


# Unidade VI:

## Ordenação Interna - Mergesort



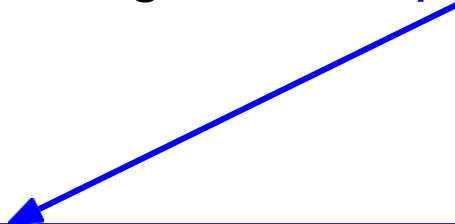
**PUC Minas**

- . Funcionamento básico
- . Algoritmo em C#
- . Análise dos número de comparações e movimentações
- . Conclusão

- **Funcionamento básico** 
- Algoritmo em C#
- Análise dos número de comparações e movimentações
- Conclusão

- . Ordenação por intercalação
- . Algoritmo de ordenação do tipo dividir para conquistar
- . Normalmente, implementado de forma recursiva e demandando um espaço adicional de memória (não é um algoritmo *in-place*)

- . Ordenação por intercalação
- . Algoritmo de ordenação do tipo dividir para conquistar
- . Normalmente, implementado de forma recursiva e demandando um espaço adicional de memória (não é um algoritmo in-place)



Um **algoritmo** de ordenação é ***in-place*** se a memória adicional requerida é independente do tamanho do *array*

# Funcionamento Básico

- Dividir sistematicamente o *array* em *subarrays* até que os mesmos tenham tamanho um
- Conquistar através da intercalação (ordenada) sistemática de dois em dois *subarrays*

# Exercício Resolvido (1)

- Faça a intercalação (ordenada) dos dois vetores [1 2 3 4 9] e [3 5 6 7 8]

## Exercício Resolvido (1)

• Faça a intercalação (ordenada) dos dois vetores [1 2 3 4 9] e [3 5 6 7 8]

[]





# Exercício Resolvido (1)

• Faça a intercalação (ordenada) dos dois vetores [4 2 3 4 9] e [3 5 6 7 8]

[]

[ 1 ]



# Exercício Resolvido (1)

• Faça a intercalação (ordenada) dos dois vetores ~~[1 2 3 4 9]~~ e [3 5 6 7 8]

[]

[ 1 ]

[ 1 2 ]



# Exercício Resolvido (1)

• Faça a intercalação (ordenada) dos dois vetores ~~[1 2 3 4 9]~~ e [3 5 6 7 8]

[]

[ 1 ]

[ 1 2 ]

[ 1 2 3 ]



# Exercício Resolvido (1)

.Faça a intercalação (ordenada) dos dois vetores ~~[1 2 3 4 9]~~ e [3 5 6 7 8]

[]

[ 1 ]

[ 1 2 ]

[ 1 2 3 ]

[ 1 2 3 3 ]



# Exercício Resolvido (1)

.Faça a intercalação (ordenada) dos dois vetores ~~[1 2 3 4 9]~~ e [3 5 6 7 8]

[]

[ 1 ]

[ 1 2 ]

[ 1 2 3 ]

[ 1 2 3 3 ]

[ 1 2 3 3 4 ]



# Exercício Resolvido (1)

• Faça a intercalação (ordenada) dos dois vetores ~~[1 2 3 4 9]~~ e ~~[3 5 6 7 8]~~

[ ]

[ 1 ]

[ 1 2 ]

[ 1 2 3 ]

[ 1 2 3 3 ]

[ 1 2 3 3 4 ]

[ 1 2 3 3 4 5 ]



## Exercício Resolvido (1)

• Faça a intercalação (ordenada) dos dois vetores ~~[1 2 3 4 9]~~ e ~~[3 5 6 7 8]~~

[]

[ 1 ]

[ 1 2 ]

[ 1 2 3 ]

[ 1 2 3 3 ]

[ 1 2 3 3 4 ]

[ 1 2 3 3 4 5 ]

[ 1 2 3 3 4 5 6 ]



## Exercício Resolvido (1)

• Faça a intercalação (ordenada) dos dois vetores ~~[1 2 3 4 9]~~ e ~~[3 5 6 7 8]~~

[]

[ 1 ]

[ 1 2 ]

[ 1 2 3 ]

[ 1 2 3 3 ]

[ 1 2 3 3 4 ]

[ 1 2 3 3 4 5 ]

[ 1 2 3 3 4 5 6 ]

[ 1 2 3 3 4 5 6 7 ]





## Exercício Resolvido (1)

• Faça a intercalação (ordenada) dos dois vetores ~~[1 2 3 4 9]~~ e ~~[3 5 6 7 8]~~

[]

[ 1 ]

[ 1 2 ]

[ 1 2 3 ]

[ 1 2 3 3 ]

[ 1 2 3 3 4 ]

[ 1 2 3 3 4 5 ]

[ 1 2 3 3 4 5 6 ]

[ 1 2 3 3 4 5 6 7 ]

[ 1 2 3 3 4 5 6 7 8 ]



## Exercício Resolvido (1)

• Faça a intercalação (ordenada) dos dois vetores ~~[1 2 3 4 9]~~ e ~~[3 5 6 7 8]~~

[]

[ 1 ]

[ 1 2 ]

[ 1 2 3 ]

[ 1 2 3 3 ]

[ 1 2 3 3 4 ]

[ 1 2 3 3 4 5 ]

[ 1 2 3 3 4 5 6 ]

[ 1 2 3 3 4 5 6 7 ]

[ 1 2 3 3 4 5 6 7 8 ]

[ 1 2 3 3 4 5 6 7 8 9 ]



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----



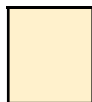
Dividir



Conquistar (intercalação ou *merge*)



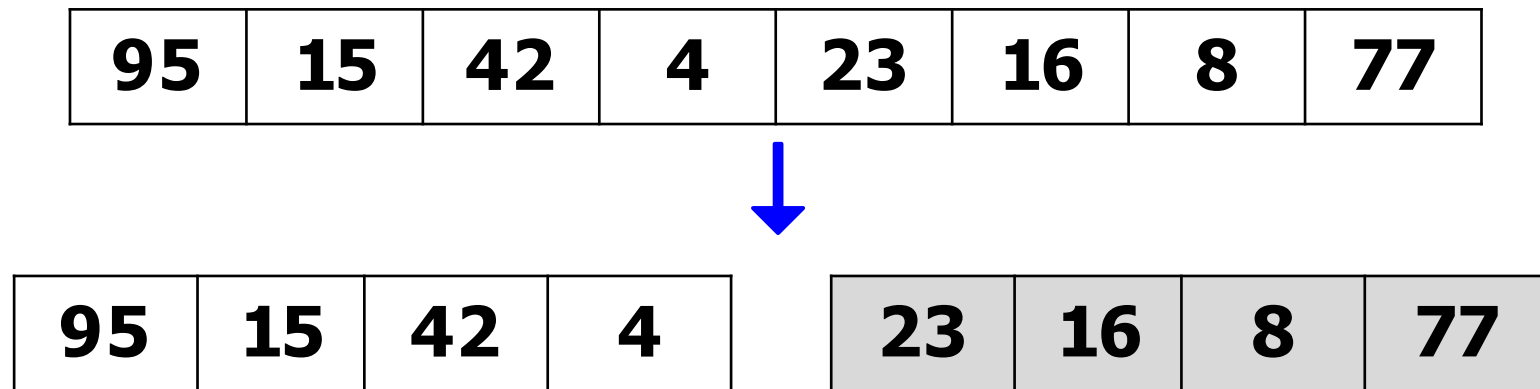
*Subarray* aguardando divisão



Elemento a ser intercalado no *subarray*

<b>95</b>	<b>15</b>	<b>42</b>	<b>4</b>	<b>23</b>	<b>16</b>	<b>8</b>	<b>77</b>
-----------	-----------	-----------	----------	-----------	-----------	----------	-----------





95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----



95	15	42	4
----	----	----	---

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

95	15
----	----

42	4
----	---





95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

95	15
----	----

42	4
----	---



95	15
----	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

--	--

42	4
----	---



95	15
----	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

15	
----	--

42	4
----	---

95	<del>15</del>
----	---------------



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

15	95
----	----

42	4
----	---

<del>95</del>	<del>15</del>
---------------	---------------



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

15	95
----	----

42	4
----	---

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

15	95
----	----

42	4
----	---



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

15	95
----	----

42	4
----	---



42	4
----	---

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

15	95
----	----

--	--



42	4
----	---



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

15	95
----	----

4	
---	--



42	<del>4</del>
----	--------------

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

95	15	42	4
----	----	----	---

23	16	8	77
----	----	---	----

15	95
----	----

4	42
---	----

<del>42</del>	<del>4</del>
---------------	--------------



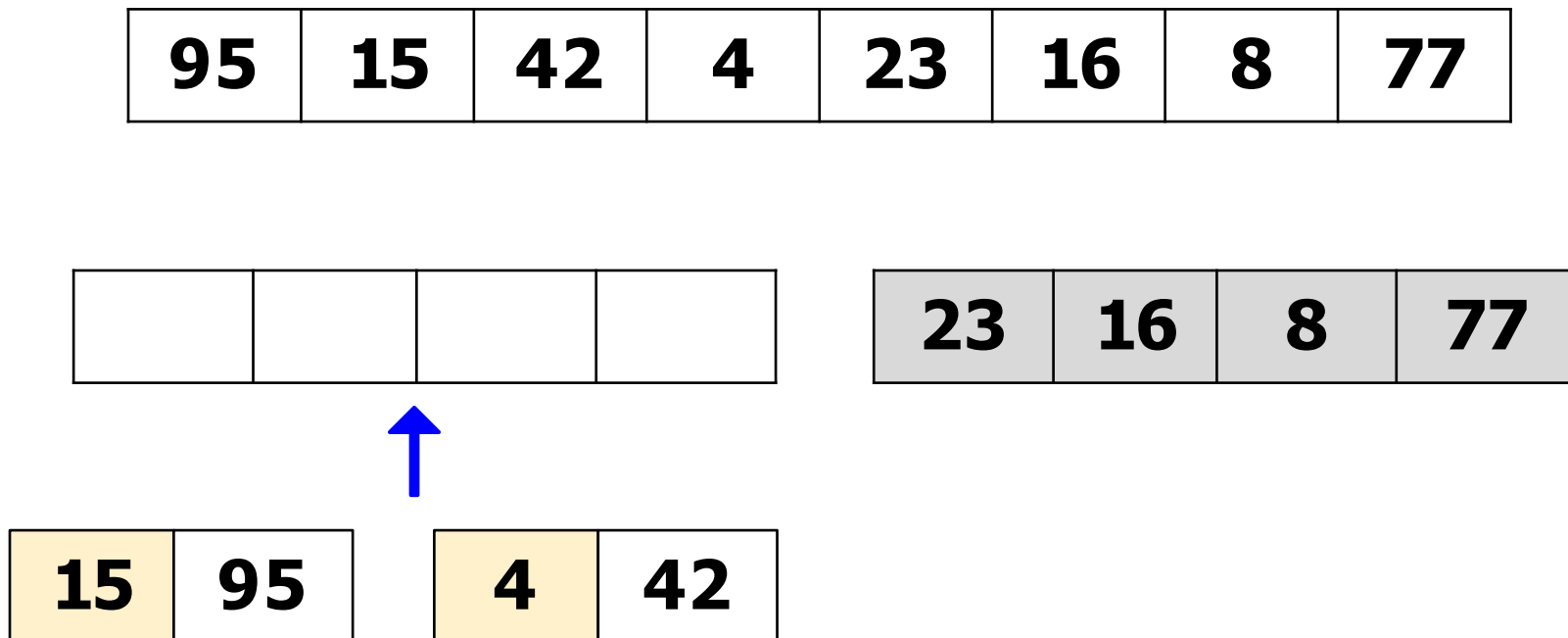
95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

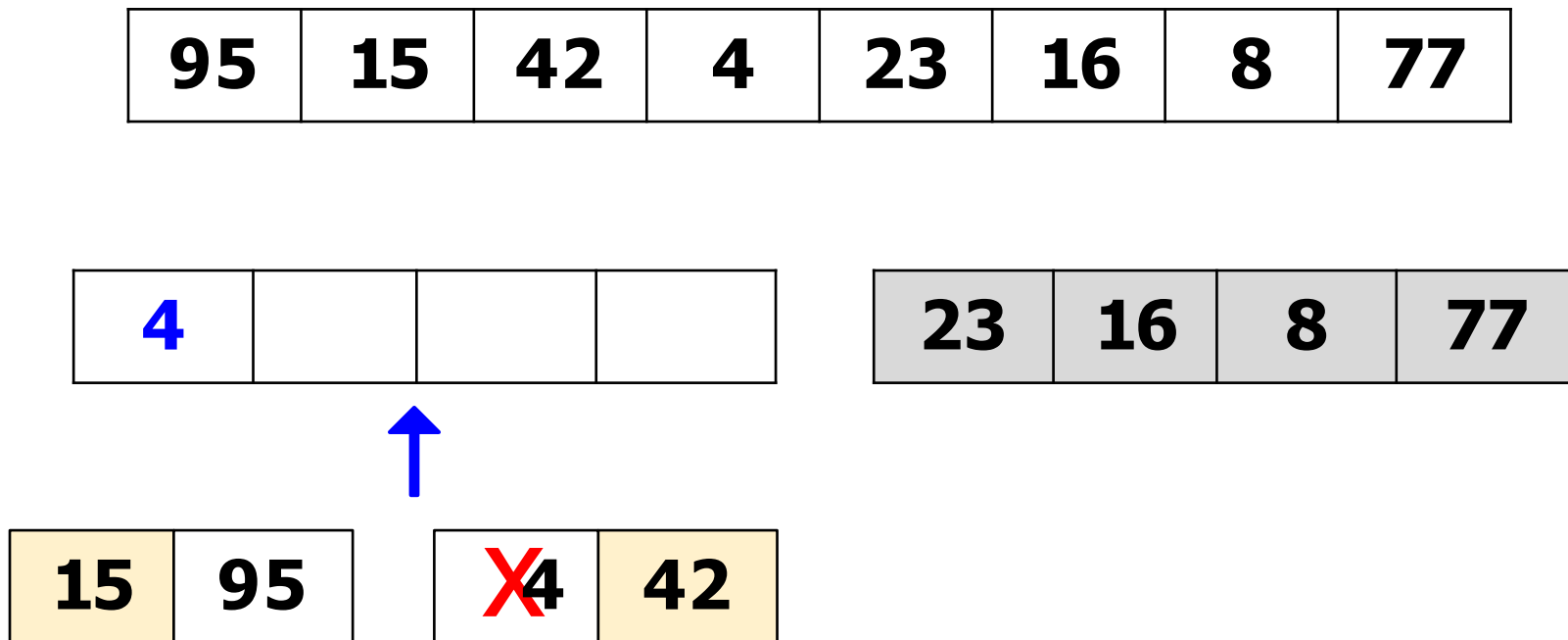
95	15	42	4
----	----	----	---

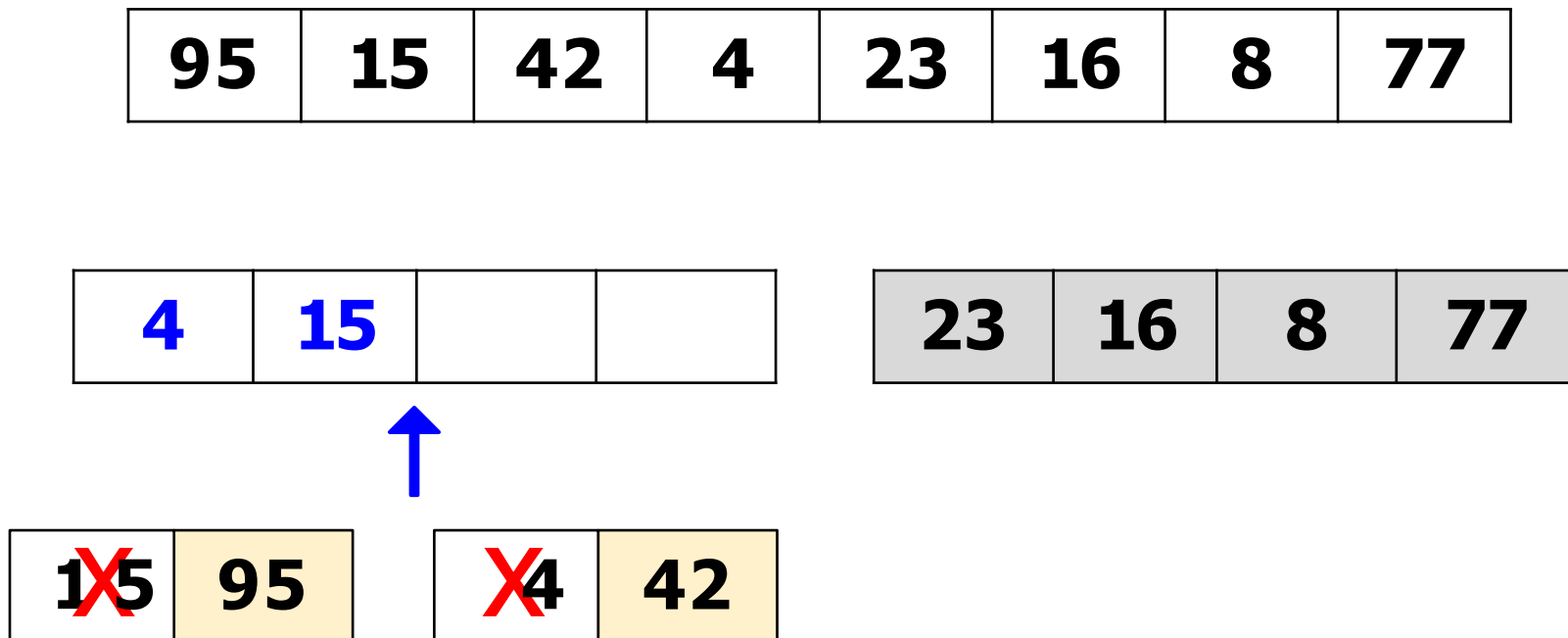
23	16	8	77
----	----	---	----

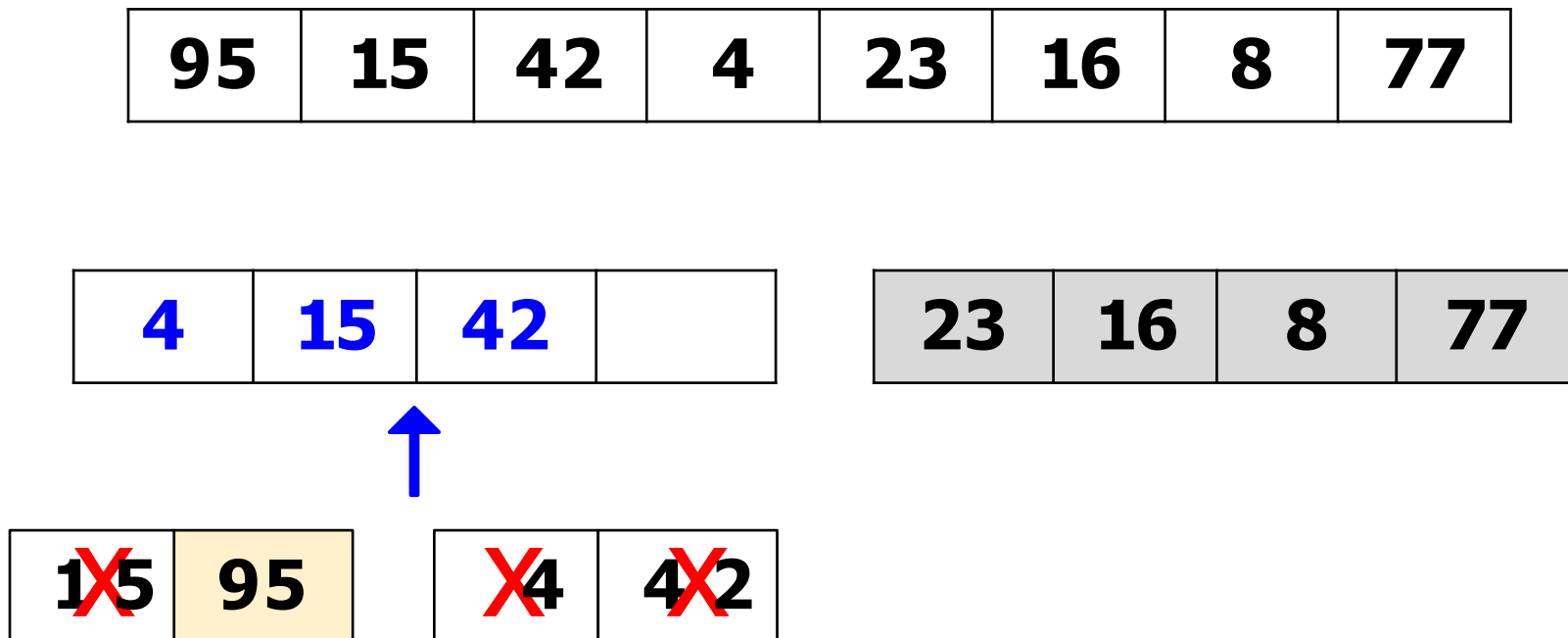
15	95
----	----

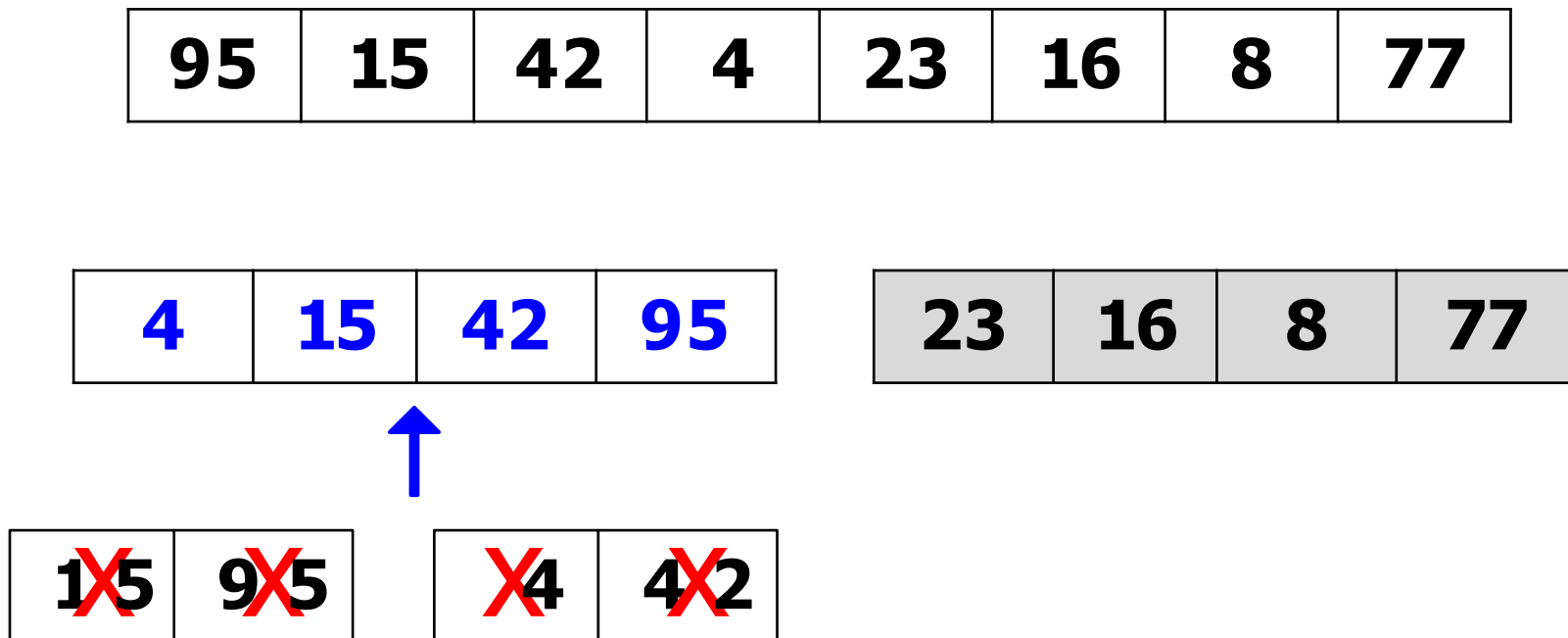
4	42
---	----













<b>95</b>	<b>15</b>	<b>42</b>	<b>4</b>	<b>23</b>	<b>16</b>	<b>8</b>	<b>77</b>
-----------	-----------	-----------	----------	-----------	-----------	----------	-----------

<b>4</b>	<b>15</b>	<b>42</b>	<b>95</b>
----------	-----------	-----------	-----------

<b>23</b>	<b>16</b>	<b>8</b>	<b>77</b>
-----------	-----------	----------	-----------

<b>95</b>	<b>15</b>	<b>42</b>	<b>4</b>	<b>23</b>	<b>16</b>	<b>8</b>	<b>77</b>
-----------	-----------	-----------	----------	-----------	-----------	----------	-----------

<b>4</b>	<b>15</b>	<b>42</b>	<b>95</b>
----------	-----------	-----------	-----------

<b>23</b>	<b>16</b>	<b>8</b>	<b>77</b>
-----------	-----------	----------	-----------



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----



23	16	8	77
----	----	---	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

23	16
----	----

8	77
---	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

23	16
----	----

8	77
---	----



23	16
----	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

--	--

8	77
---	----



23	16
----	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

16	
----	--

8	77
---	----



23	<del>16</del>
----	---------------

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

16	23
----	----

8	77
---	----



<del>23</del>	<del>16</del>
---------------	---------------



<b>95</b>	<b>15</b>	<b>42</b>	<b>4</b>	<b>23</b>	<b>16</b>	<b>8</b>	<b>77</b>
-----------	-----------	-----------	----------	-----------	-----------	----------	-----------

<b>4</b>	<b>15</b>	<b>42</b>	<b>95</b>
----------	-----------	-----------	-----------

<b>23</b>	<b>16</b>	<b>8</b>	<b>77</b>
-----------	-----------	----------	-----------

<b>16</b>	<b>23</b>
-----------	-----------

<b>8</b>	<b>77</b>
----------	-----------

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

16	23
----	----

8	77
---	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

16	23
----	----

8	77
---	----



8	77
---	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

16	23
----	----

--	--

8	77
---	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

16	23
----	----

8	
---	--

<del>8</del>	77
--------------	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----


4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

16	23
----	----

8	77
---	----

<del>8</del>	<del>77</del>
--------------	---------------



## Exemplo

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

23	16	8	77
----	----	---	----

16	23
----	----

8	77
---	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

--	--	--	--



16	23
----	----

8	77
---	----



95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

8			
---	--	--	--



16	23
----	----

<del>8</del>	77
--------------	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

8	16		
---	----	--	--



<del>16</del>	23
---------------	----

<del>8</del>	77
--------------	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

8	16	23	
---	----	----	--



<del>1</del> 6	<del>2</del> 3
----------------	----------------

<del>X</del> 8	77
----------------	----

95	15	42	4	23	16	8	77
----	----	----	---	----	----	---	----

4	15	42	95
---	----	----	----

8	16	23	77
---	----	----	----

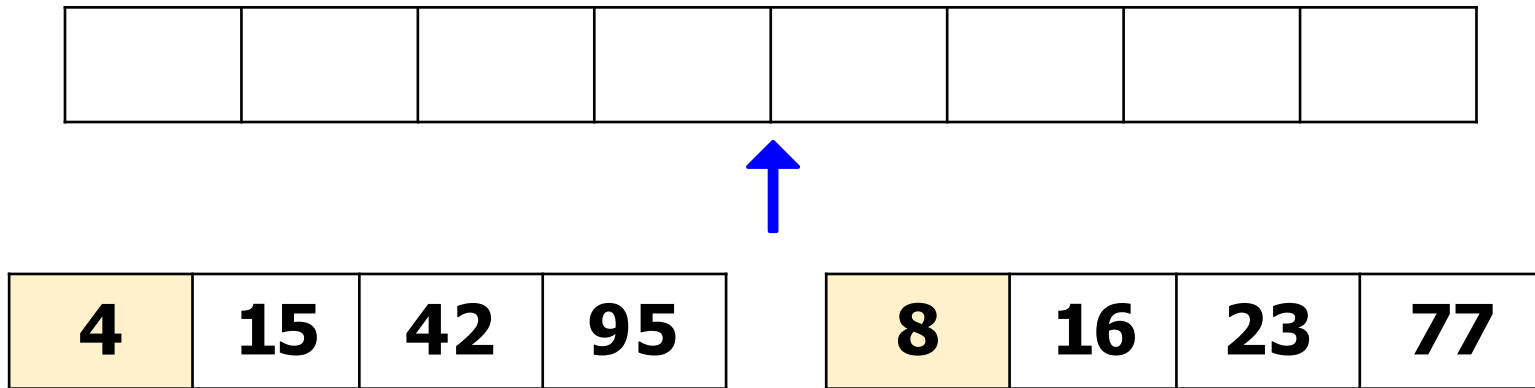


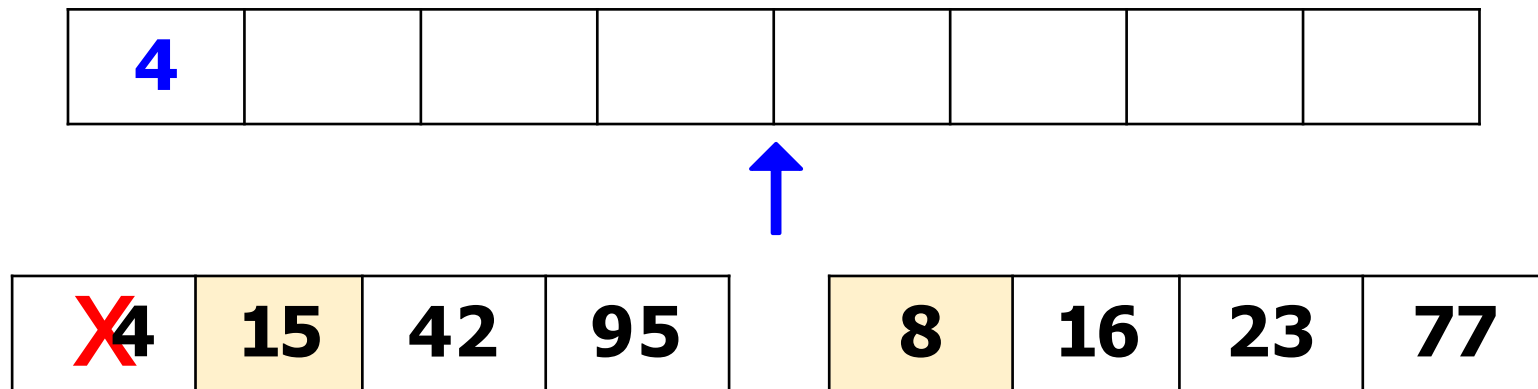
<del>1</del> 6	<del>2</del> 3	<del>X</del> 8	<del>7</del> 7
----------------	----------------	----------------	----------------

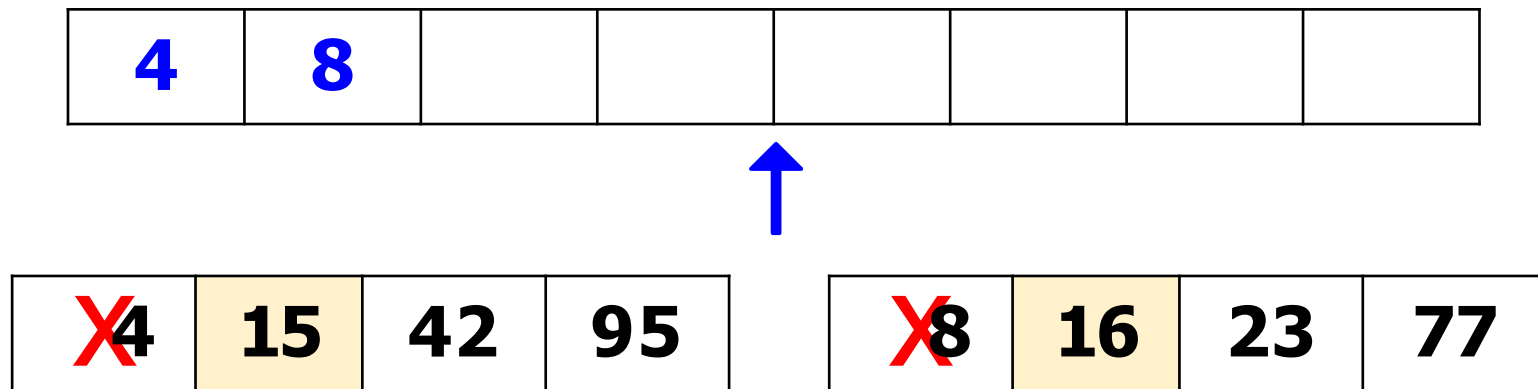
<b>95</b>	<b>15</b>	<b>42</b>	<b>4</b>	<b>23</b>	<b>16</b>	<b>8</b>	<b>77</b>
-----------	-----------	-----------	----------	-----------	-----------	----------	-----------

<b>4</b>	<b>15</b>	<b>42</b>	<b>95</b>
----------	-----------	-----------	-----------

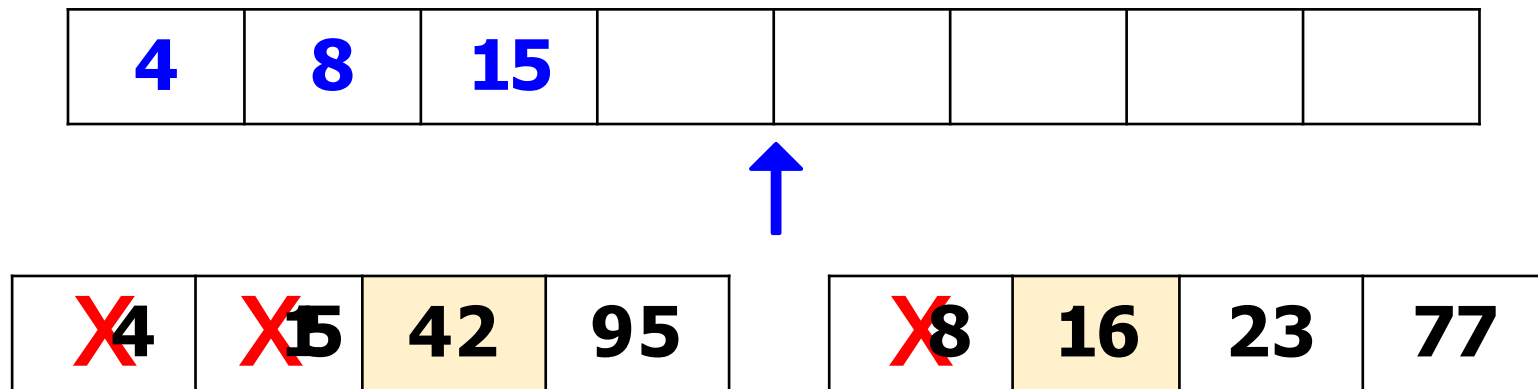
<b>8</b>	<b>16</b>	<b>23</b>	<b>77</b>
----------	-----------	-----------	-----------

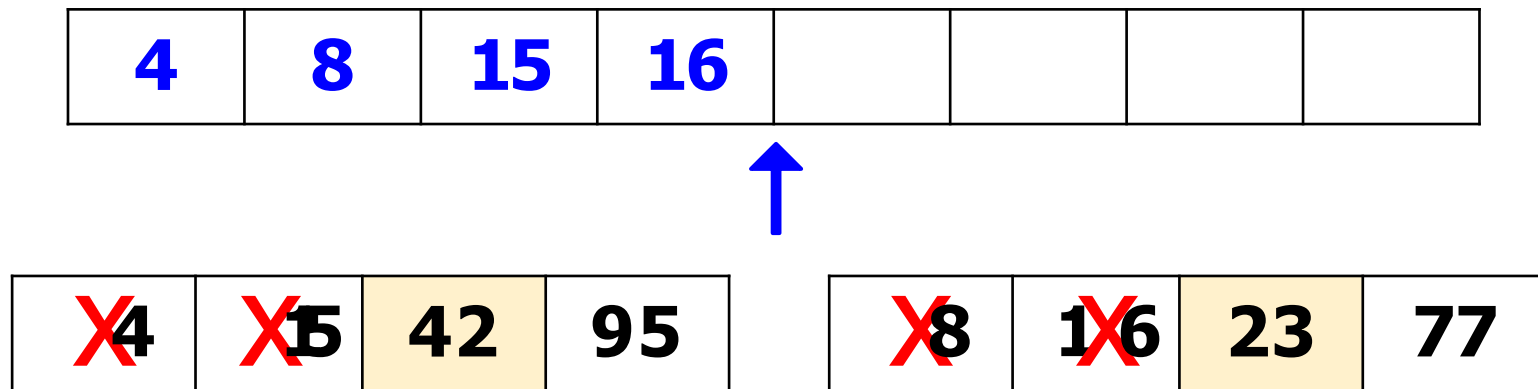


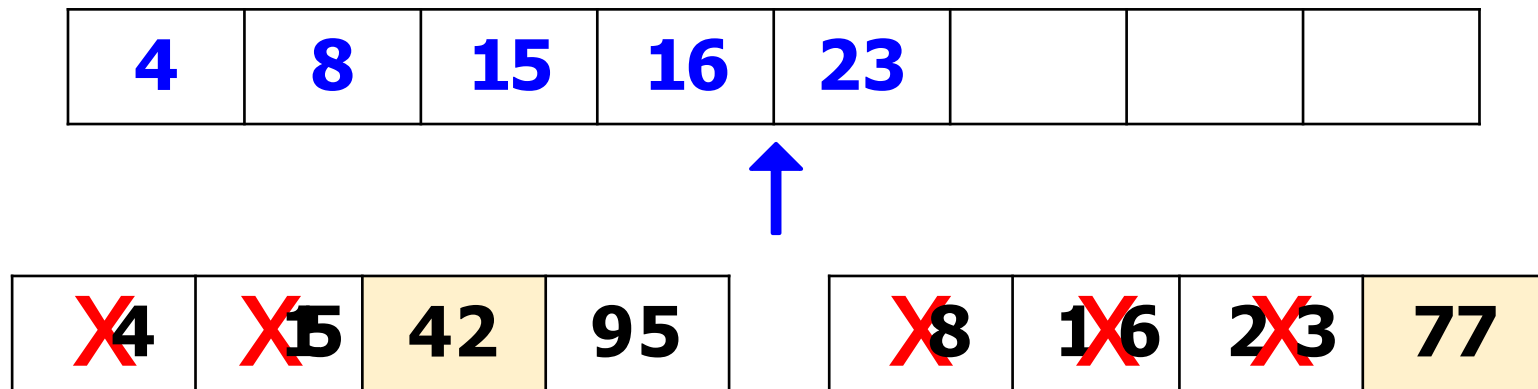


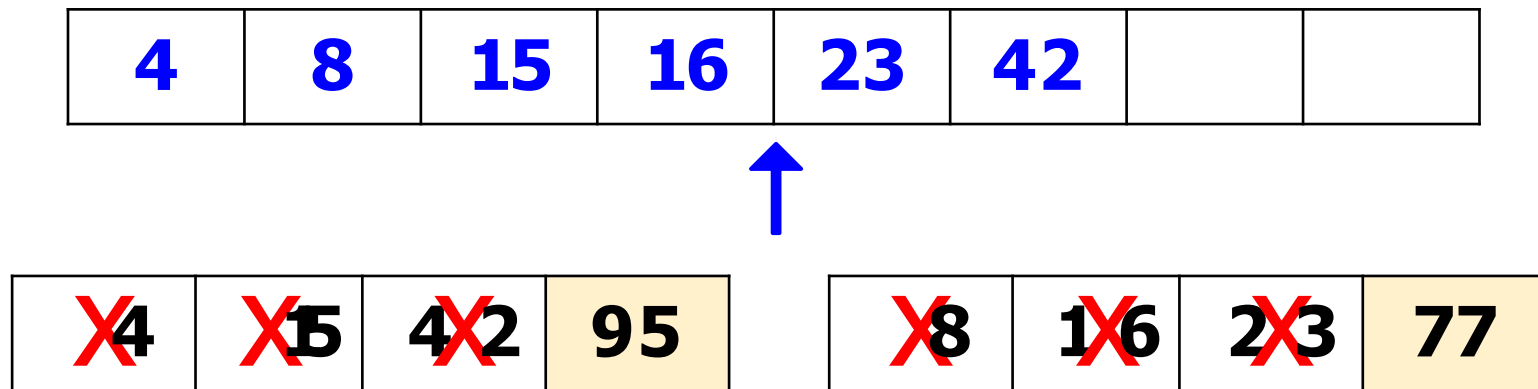


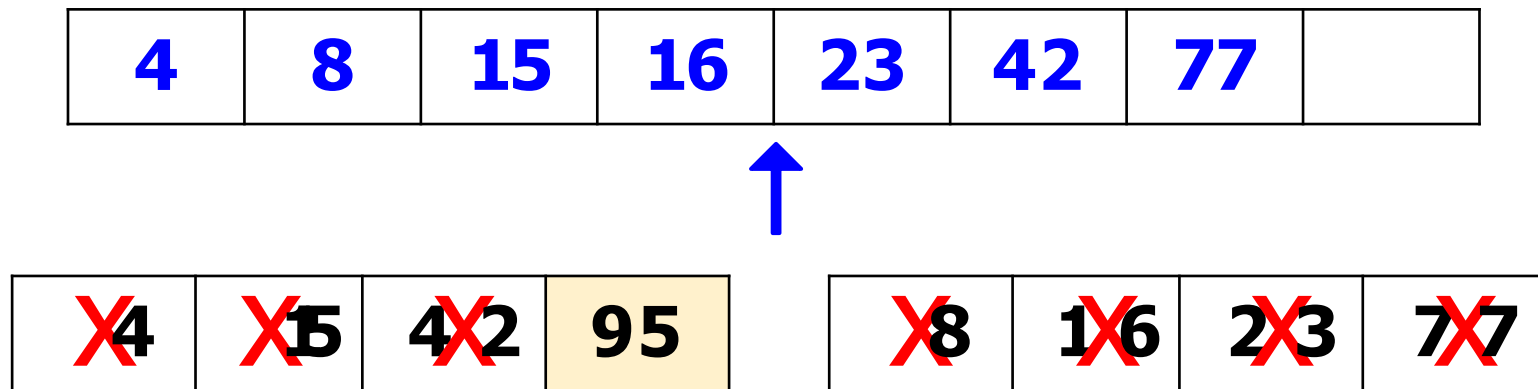


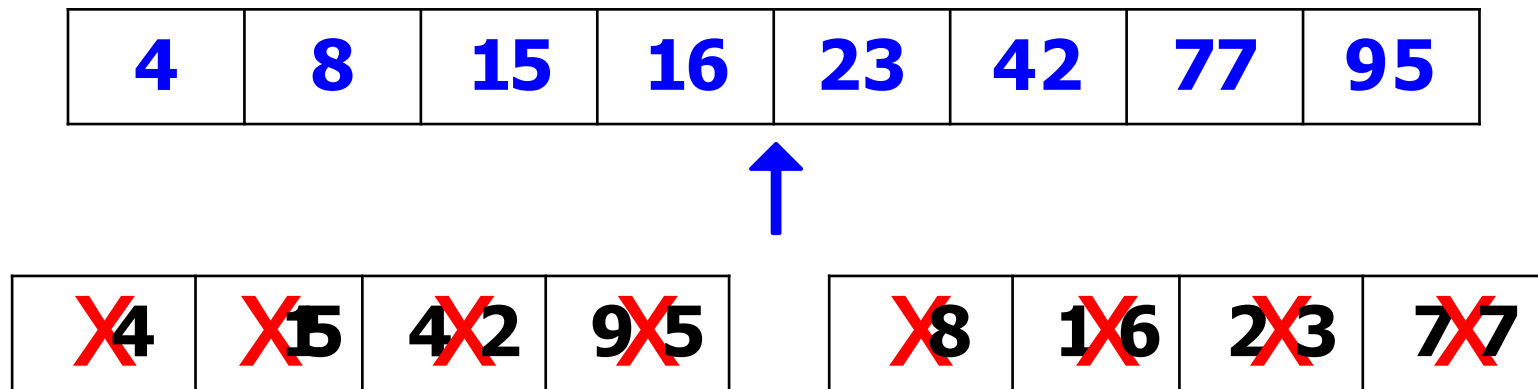













<b>4</b>	<b>8</b>	<b>15</b>	<b>16</b>	<b>23</b>	<b>42</b>	<b>77</b>	<b>95</b>
----------	----------	-----------	-----------	-----------	-----------	-----------	-----------

- . Funcionamento básico
- . **Algoritmo em C#** 
- . Análise dos número de comparações e movimentações
- . Conclusão



## Algoritmo em C#

```
void MergeSort(int [] v, int esq, int dir) {  
    if (esq < dir){  
        int meio = (esq + dir) / 2;  
        MergeSort(v, esq, meio);  
        MergeSort(v, meio + 1, dir);  
        Intercala(v, esq, meio, dir);  
    }  
}
```

## Algoritmo em C#

```
void Intercala(int[] v, int esq, int meio, int dir){  
    int n1 = meio - esq + 1;  
    int n2 = dir - meio;  
  
    int[] v_esq = new int[n1];  
    int[] v_dir = new int[n2];  
  
    for (int i = 0; i < n1; i++)  
        v_esq[i] = v[esq + i];  
    for (int j = 0; j < n2; j++)  
        v_dir[j] = v[meio + 1 + j];  
  
    int cont1 = 0, cont2 = 0;  
    int k = esq;
```

## Algoritmo em C#

```
while (cont1 < n1 && cont2 < n2){  
    if (v_esq[cont1] <= v_dir[cont2]){  
        v[k] = v_esq[cont1];  
        cont1++;  
    }  
    else{  
        v[k] = v_dir[cont2];  
        cont2++;  
    }  
    k++;  
}
```

## Algoritmo em C#

```
while (cont1 < n1){  
    v[k] = v_esq[cont1];  
    cont1++;  
    k++;  
}
```

```
while (cont2 < n2){  
    v[k] = v_dir[cont2];  
    cont2++;  
    k++;  
}
```

```
}
```

## Algoritmo em C#

```
void Intercala(int[] v, int esq, int meio, int dir){
    int n1 = meio - esq + 1;
    int n2 = dir - meio;

    int[] v_esq = new int[n1];
    int[] v_dir = new int[n2];

    for (int i = 0; i < n1; i++)
        v_esq[i] = v[esq + i];
    for (int j = 0; j < n2; j++)
        v_dir[j] = v[meio + 1 + j];


    int cont1 = 0, cont2 = 0;
    int k = esq;

    while (cont1 < n1 && cont2 < n2){
        if (v_esq[cont1] <= v_dir[cont2]){
            v[k] = v_esq[cont1];
            cont1++;
        }
    }
```

```
    else{
        v[k] = v_dir[cont2];
        cont2++;
    }
    k++;
}

while (cont1 < n1){
    v[k] = v_esq[cont1];
    cont1++;
    k++;
}

while (cont2 < n2){
    v[k] = v_dir[cont2];
    cont2++;
    k++;
}
}
```

- Funcionamento básico
- Algoritmo em C#
- **Análise dos número de comparações e movimentações** 
- Conclusão

# Análise do Número de Comparações

. Todos os casos:

- Em cada *subarray* (tamanho  $k$ ), fazemos  $k - 1$  comparações
- Supondo que o tamanho do *array* é uma potência de 2, fazemos  $\lg(n)$  passos

$$\left\{ \begin{array}{l} C(1) = 0 \\ C(n) = 2C(n/2) + \Theta(n) \end{array} \right. \quad \Theta(n * \lg(n))$$


# Análise do Número de Movimentações

. Todos os casos:

- Movimentamos os elementos de cada subarray duas vezes

$$\begin{cases} M(1) = 0 \\ M(n) = 2M(n/2) + \Theta(n) \end{cases} \quad \Theta(n \lg(n))$$



- . Funcionamento básico
- . Algoritmo em C#
- . Análise dos número de comparações e movimentações
- . **Conclusão** 

# Conclusão

- Método estável
- Normalmente, implementado de forma recursiva e demandando memória adicional
- Faz  $\Theta(n \lg n)$  comparações nos três casos (melhor, médio e pior)

## Exercício (1)

.Mostre todas as comparações e movimentações do algoritmo anterior para o *array* abaixo:

12	4	8	2	14	17	6	18	10	16	15	5	13	9	1	11	7	3
----	---	---	---	----	----	---	----	----	----	----	---	----	---	---	----	---	---